



universität  
wien



# Fast Rerouting Against Dynamic Failures: 2-Resilience via Ear- Decomposition and Planarity

Wenkai Dai (TU Berlin, Germany & University of Vienna, Austria), Klaus-Tycho Foerster (TU Dortmund, Germany), and Stefan Schmid (TU Berlin, Germany)

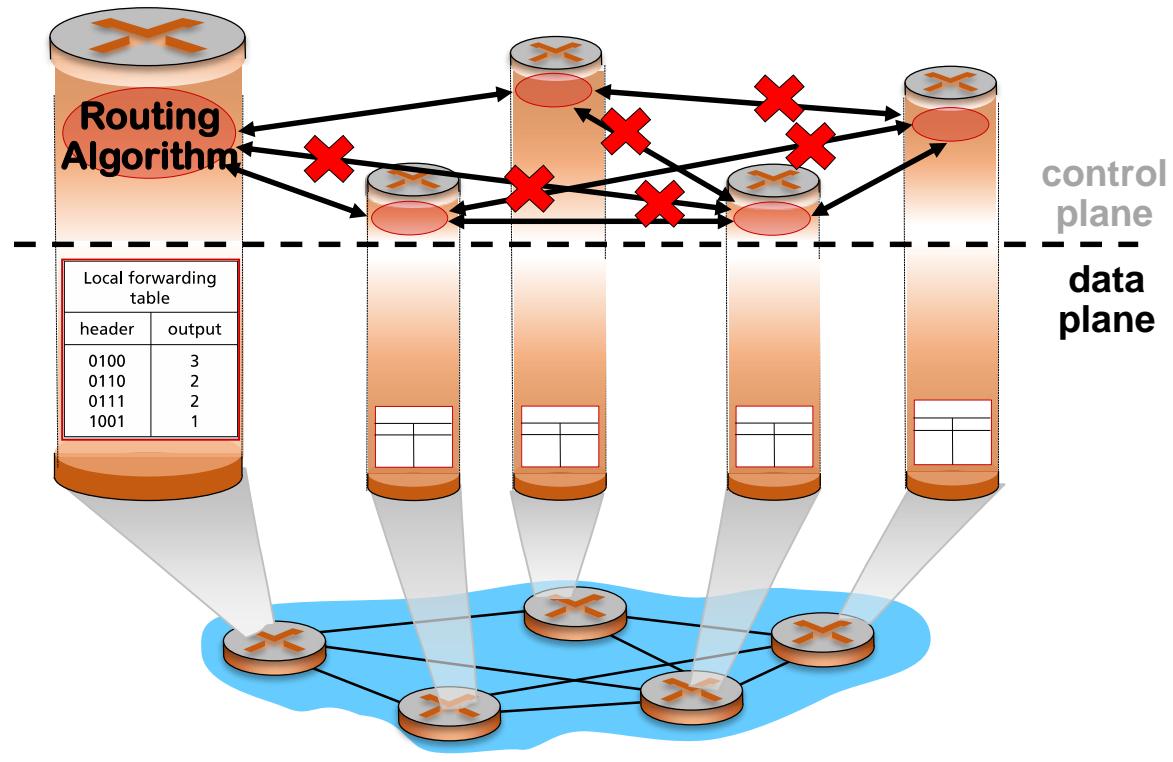


29th Conference on Principles of Distributed Systems



# Motivation: Handling Link-Failures on Control-Plane Is Slow

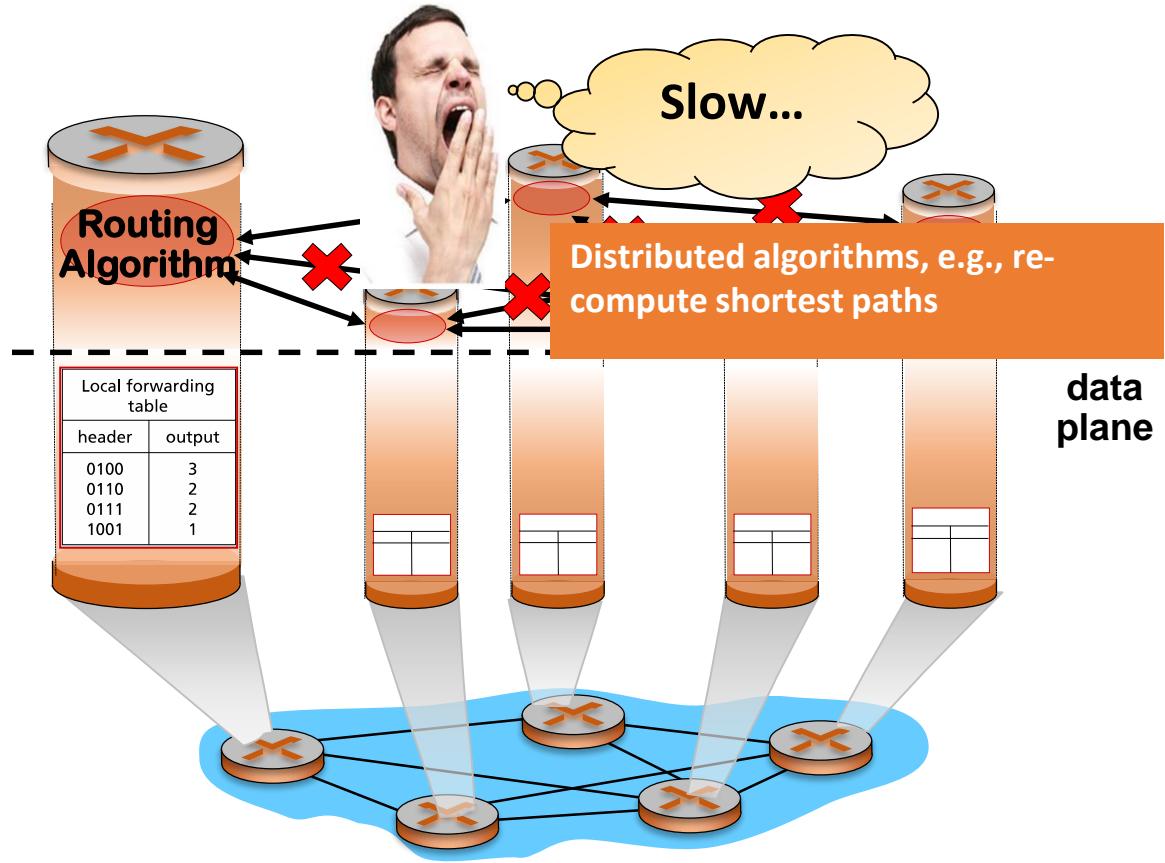
- Link **failures** are **unavoidable** so mechanism to **route around**
- Traditionally, control-plane handles link failures by recomputing routing tables (e.g., OSPF)
- **Slow**, unacceptable for critical applications



(Credits: Stefan Schmid)

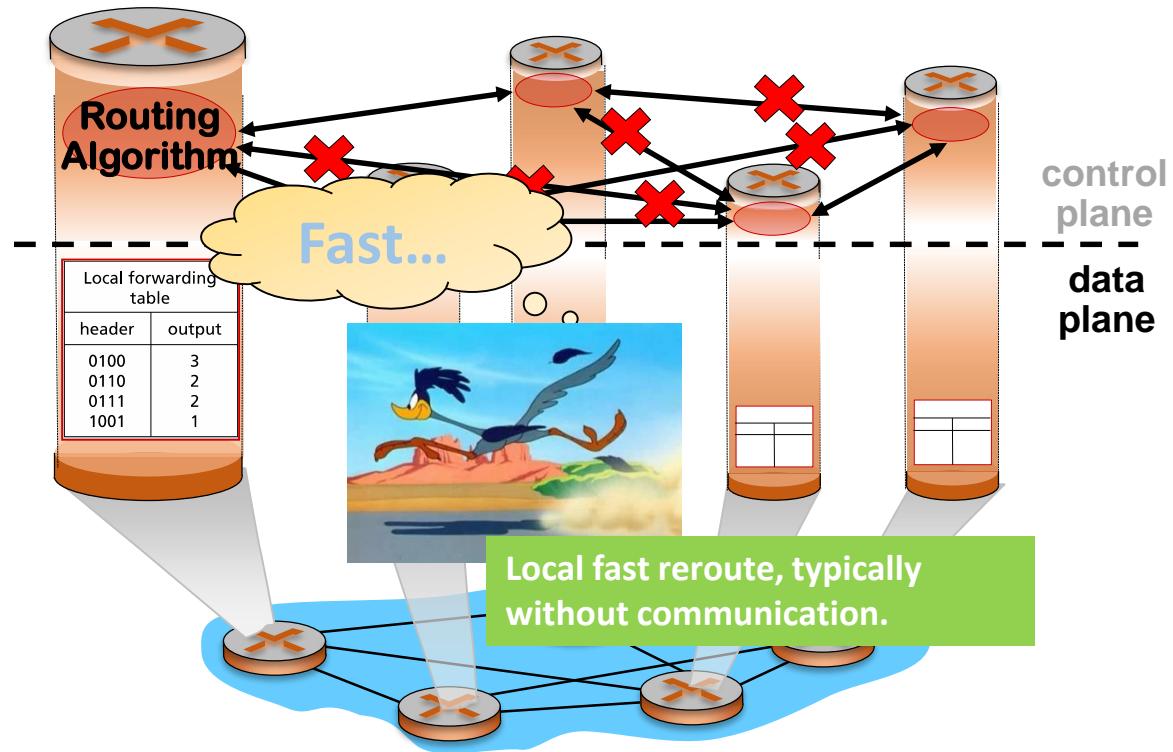
# Motivation: Handling Link-Failures on Control-Plane Is Slow

- Link **failures** are **unavoidable** so mechanism to **route** around
- Traditionally, control-plane handles link failures by recomputing routing tables (e.g., OSPF)
- **Slow**, unacceptable for critical applications



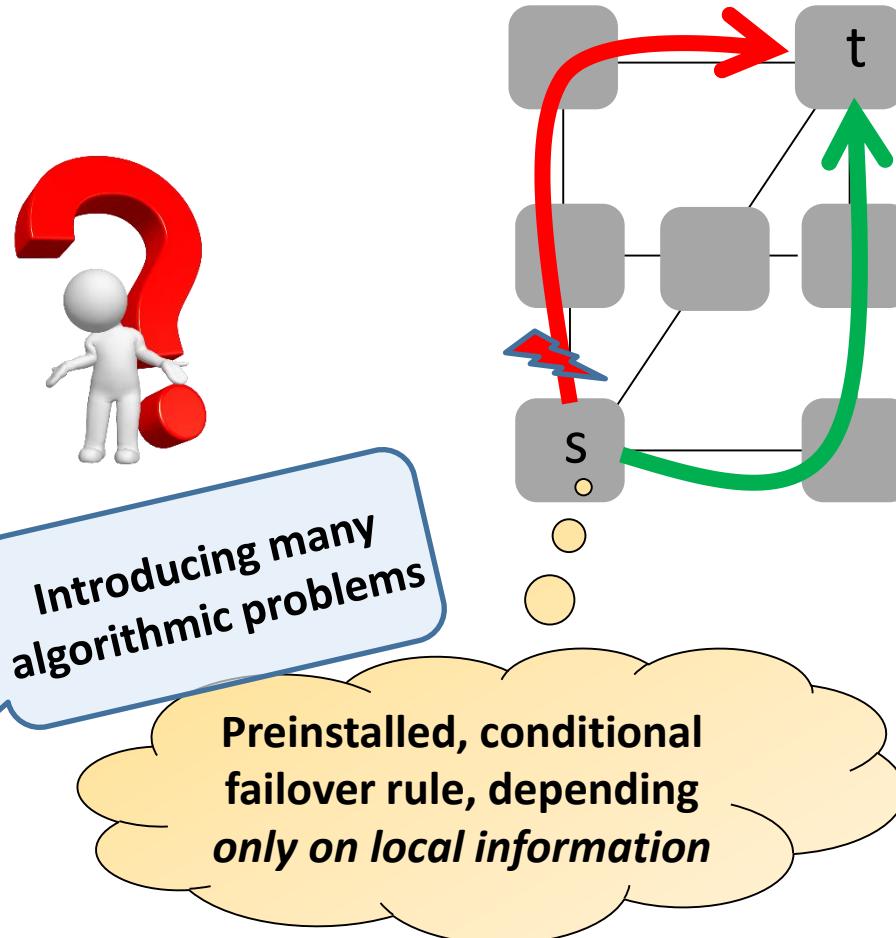
# Motivation: Handling Link-Failures on Control-Plane Is Slow

- Fast reroute (FRR) on data-plane handles failures locally



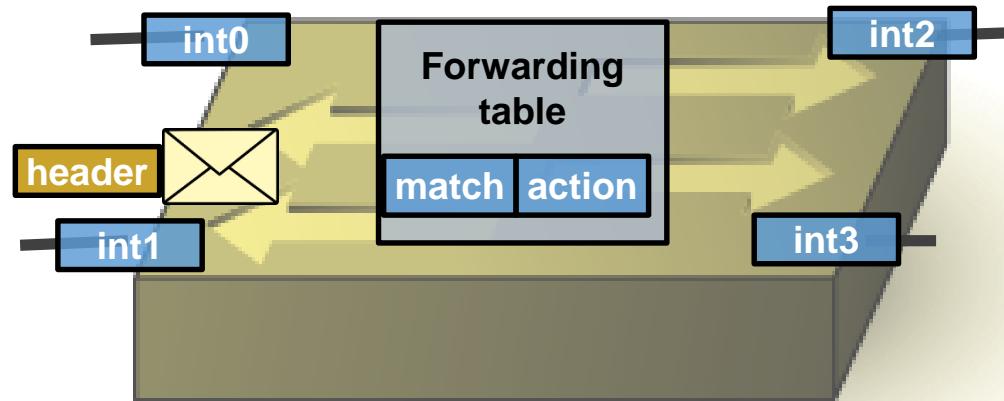
# Fast Rerouting (FRR) on Data-Plane

- Local failover without invoking control plane, by pre-installing alternative paths
- Conditional forwarding rules, only depending on local failures, no information about failures downstream
- Challenge: How to determine these pre-installing reroute rules



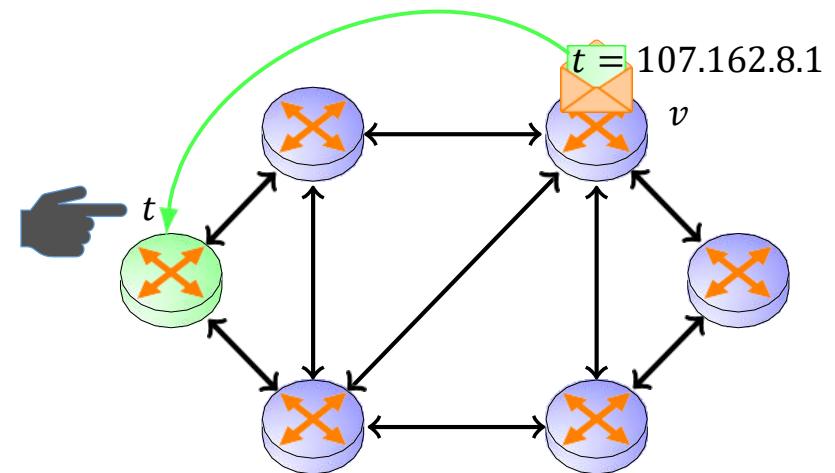
# Basic Routing Functions Only Use Local Information

- Network is a connected undirected graph  $G=(V,E)$
- Packet **forwarding** can be model as **a function**



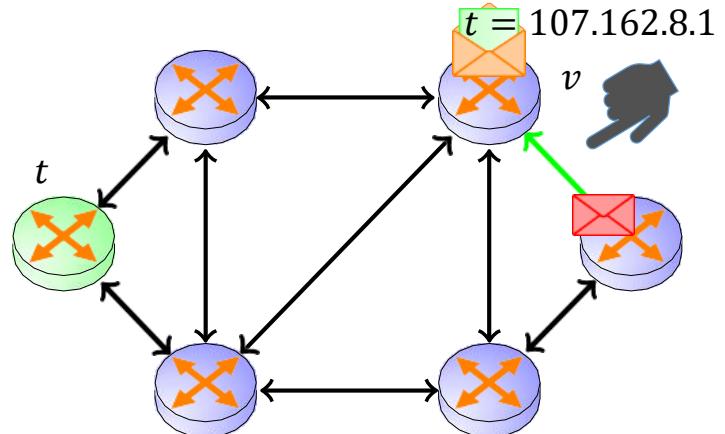
# Basic Routing Functions Only Use Local Information

- Network is a connected undirected graph  $G=(V,E)$
- **Basic** routing function at node  $v$  may only match on:
  - Destination  $t$



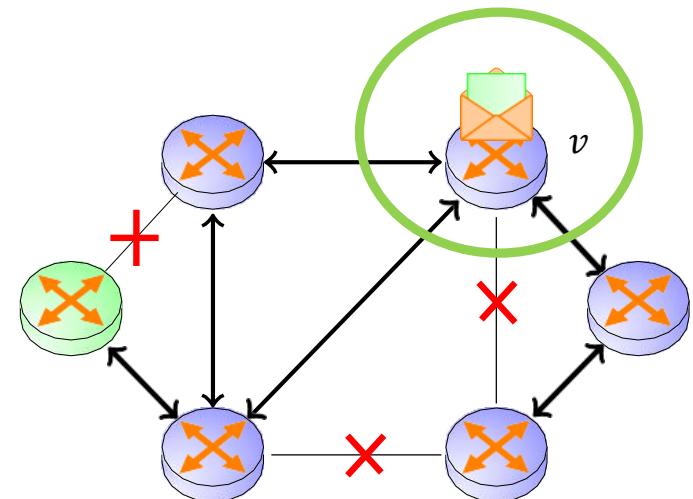
# Basic Routing Functions Only Use Local Information

- Network is a connected undirected graph  $G=(V,E)$
- **Basic** routing function at node  $v$  may only match on:
  - Destination  $t$
  - Incoming port from  $E(v) \cup \{\perp\}$



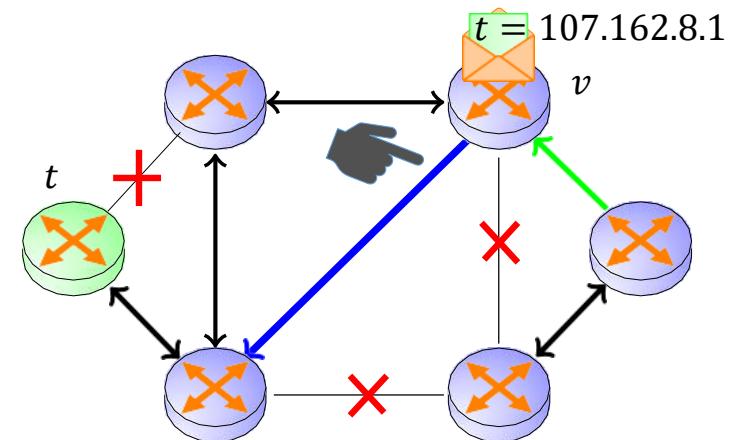
# Basic Routing Functions Only Use Local Information

- Network is a connected undirected graph  $G=(V,E)$
- **Basic** routing function at node  $v$  may only match on:
  - Destination  $t$
  - Incoming port from  $E(v) \cup \{\perp\}$
  - Incident edge failures  $F \cap E(v)$



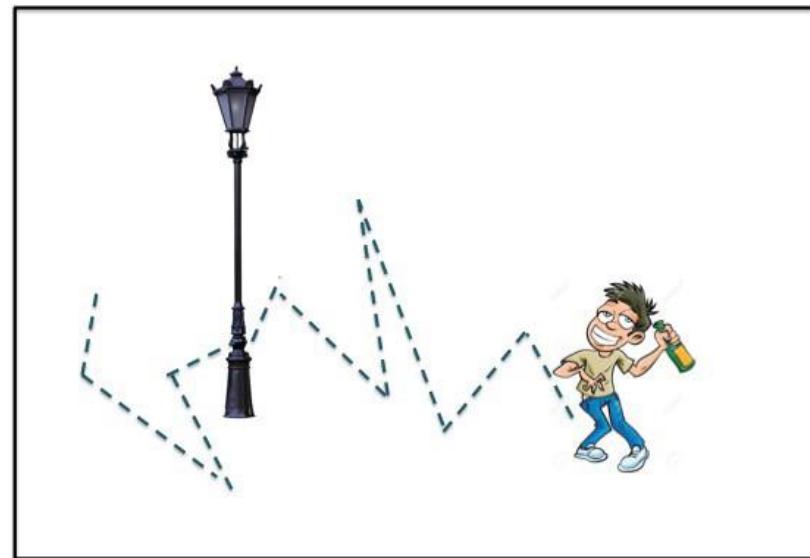
# Basic Routing Functions Only Use Local Information

- Network is a connected undirected graph  $G=(V,E)$
- **Basic** routing function at node  $v$  may only match on:
  - Destination  $t$
  - Incoming port from  $E(v) \cup \{\perp\}$
  - Incident edge failures  $F \cap E(v)$
- To determine an outgoing link in  $E(v) - F$
- We know, e.g., how to achieve 1-resilience [Kwong et al. 2011], [Feigenbaum et al. 2012]



# Our Focus: Deterministic Forwarding Rules

- Only consider **deterministic** routing functions
- Randomized routing cons:
  - packet recording
  - long paths?
  - device limitations



# Spectrum of Models



## *Basic Routing*

Incident  
links

Per  
destination

Incoming  
port

Routing  
Function

Outgoing  
port

## *Source-matched Routing*

Per  
source

Incident  
links

Per  
destination

Incoming  
port

Routing  
Function

Outgoing  
port

## *Optional: Header Bit-rewriting*

Packet header  
rewritable-bits

Inputs of above  
routing, e.g., basic

Routing  
Function

Outgoing  
port

Packet header  
rewritable-bits

# Spectrum of Models



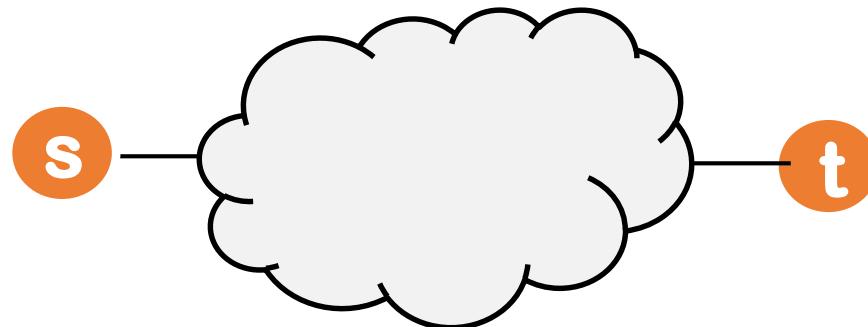
- **The more we match, the larger the tables**
- **Source-matched:  $O(n^2)$  (for source and destinations), times failure scenarios (?)**
- **Bits-rewriting: even more overheads (and where to write in header?)**

# Problem: Computing a $k$ -Resilient Routing Scheme

- Given an undirected graph  $G=(V,E)$  and a destination  $t \in V$
- A routing scheme is called  $k$ -Resilient ( $1 \leq k \leq |E|$ ) :
  - if any  $v \in V$  can reach a destination  $t \in V$  **as long as  $v-t$  still connected in  $G-F$** , for  $k$  arbitrary link failures  
 $F \subseteq E(G)$
- Goal: Compute a  $k$ -resilient routing scheme for given  $G=(V,E)$ ,  $t \in V$  and  $k$

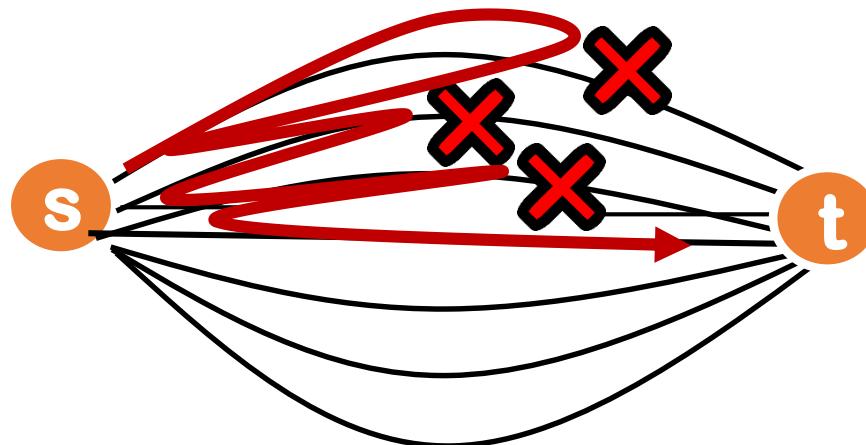
# The Model Matters

- Assume k-connected graph: how to tolerate  $k-1$  failures?
- If I can match source, import and destination?



# The Model Matters

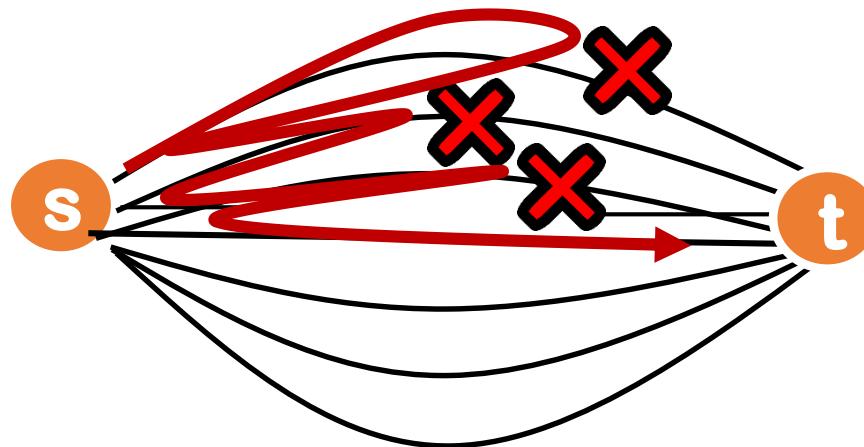
- Assume k-connected graph: how to tolerate k-1 failures?
- If I can match source, import and destination?



- Try one path after the other!

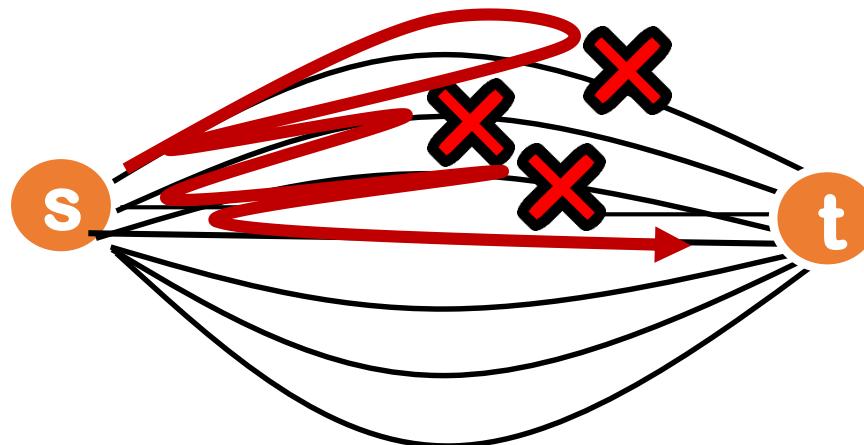
# The Model Matters

- Assume k-connected graph: how to tolerate k-1 failures?
- If I can match source, import and destination?



# The Model Matters

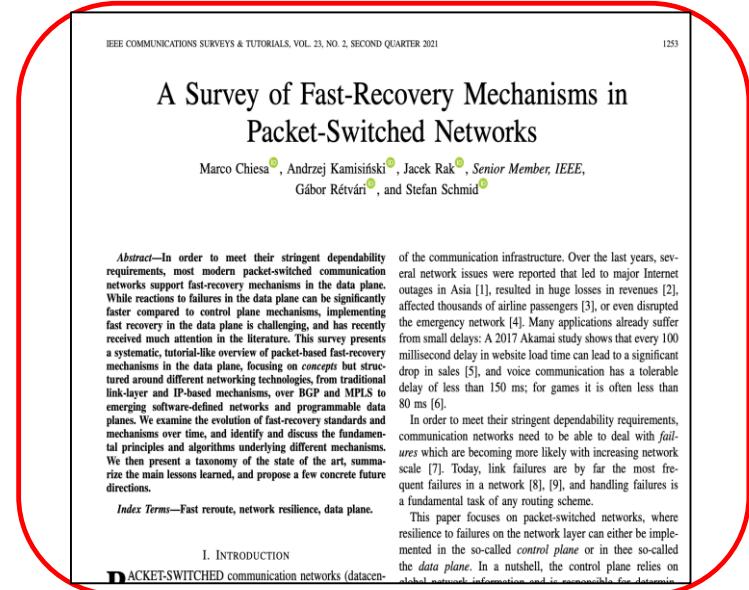
- Assume k-connected graph: how to tolerate  $k-1$  failures?
- If I can match source, import and destination?



- Long-standing conjecture (that it still works)!

# Related Works (Static Link Failures)

- Fast failover is a widely studied area, see recent survey
  - Chiesa et al. (2021): *A Survey of Fast Recovery Mechanisms in Packet-Switched Networks*
- Source-matched routing
  - Latest result: *Dai et al. (SPAA' 23): A Tight Characterization of Fast Failover Routing: Resiliency to Two Link Failures is Possible*



# The Model of Static Link Failure Is Unrealistic

- Previous research considers **static** link failures, i.e.,
  - **Simultaneous** and **permanent** (*unrecoverable*) Failures

# The Model of Static Link Failure Is Unrealistic

- Previous research considers **static** link failures, i.e.,
  - **Simultaneous** and **permanent** (*unrecoverable*) Failures

## Reality:

- Links fail *over the time*
- And can recover!

P. Gill et al. Understanding network failures in data centers:  
measurement, analysis, and implications (Sigcomm), 2011.

# The Model of Static Link Failure Is Unrealistic

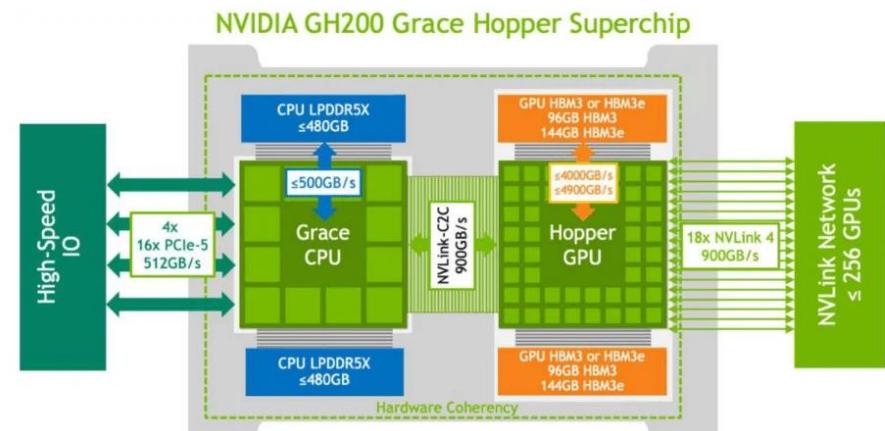
- Previous research considers **static** link failures, i.e.,
  - **Simultaneous** and **permanent (unrecoverable)** Failures

## Reality:

- Links fail *over the time*
- And can recover!

P. Gill et al. Understanding network failure modes: measurement, analysis, and implications

E.g., NVLink vs Copper



# The Novel Models for Link Failures

Challenges: devise  
failover reroute rules on  
dynamic graphs?

## Semi-Dynamic Link Failure

Links  $F$  fail *non-simultaneously*,  
but *still permanently*.

## Dynamic Link Failure

Links  $F$  fail *non-simultaneously*,  
and *non-permanently*.

First insights: Dai et al. On the Resilience of Fast  
Failover Routing Against Dynamic Link Failures (IFIP  
Networking), 2025.

# The Novel Models for Link Failures

Challenges: devise  
failover reroute rules on  
dynamic graphs?

## Semi-Dynamic Link Failure

Links  $F$  fail *non-simultaneously*,  
but *still permanently*.

## Dynamic Link Failure

Links  $F$  fail *non-simultaneously*,  
and *non-permanently*.

Algorithm that works for semi-dynamic also works for dynamic!

First insights: Dai et al. On the Resilience of Fast  
Failover Routing Against Dynamic Link Failures (IFIP  
Networking), 2025.

# Related Previous Results and Our Main Contributions

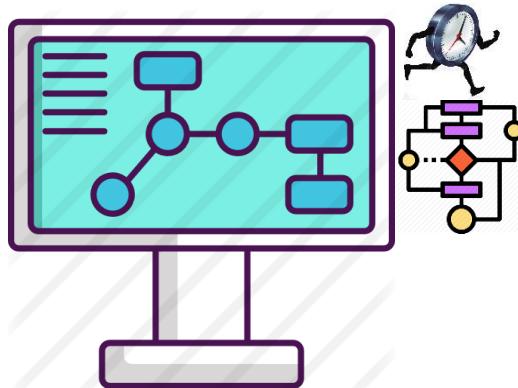
Failure type	Per-destination	Per source	Incoming port	Incident links	$k$ -edge-connected graph	Graph Restrictions	Packet rewriting bits	Resiliency	Ref.
static	X		X	X	$k = 2$	planar		No 2-resilient	[Chiesa et al. 2016]
static	X	X	X	X		general		No 3-resilient	[Dai et al. 2023]
static	X	X	X	X		general		$\leq 2$ -resilient	[Dai et al. 2023]
static	X		X	X		general	1 bit	$\geq 2$ -resilient?	Open
static	X	X	X	X		general		$\geq 2$ -resilient?	Open
dynamic	X		X	X	$k = 2$	planar	0	No 2-resilient	[This Talk]
dynamic	X		X	X		general	$\log k$	No $k$ -resilient	[This Talk]
dynamic	X		X	X		planar	1 or 2	$\leq 2$ -resilient	[This Talk]

Previous results

Our results

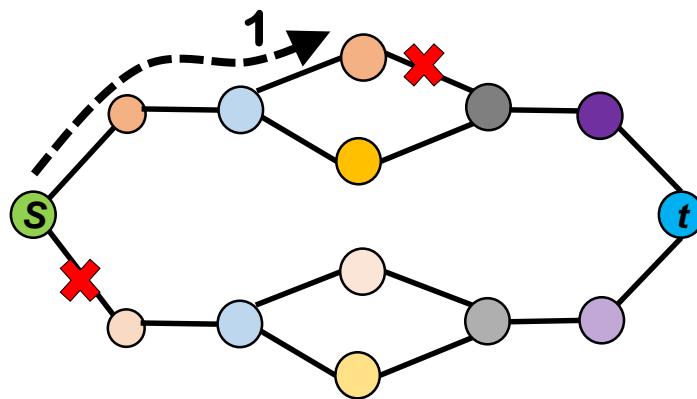
## In particular:

- A 2-resilient **basic** routing scheme against **dynamic** failures by **rewriting 1-bit (tight)** on **planar** graphs, **computable** within  $O(n + m)$  per destination
- A 2-resilient **source-matched** routing against **dynamic** failures on **planar** graphs **without** rewriting is **impossible**



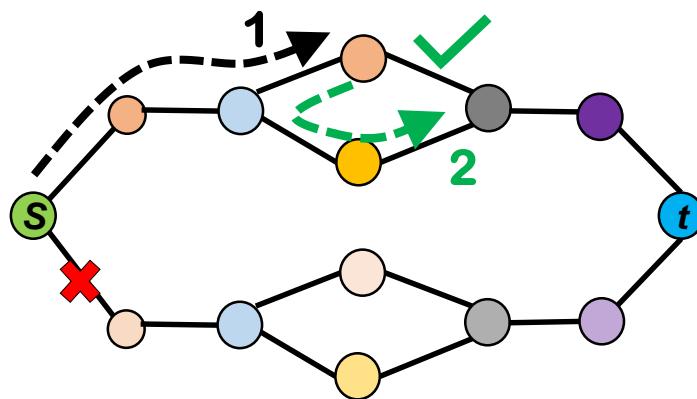
# Impossibility: Via Case Distinction

- Impossible to have 2-resilient *source-matched routing* against dynamic failures *without rewriting bits* even on planar graph
  - In the example, every node must use **link-circular routing** (clockwise or counter-clockwise)
  - Depending on order, fail dynamically: *indistinguishable*



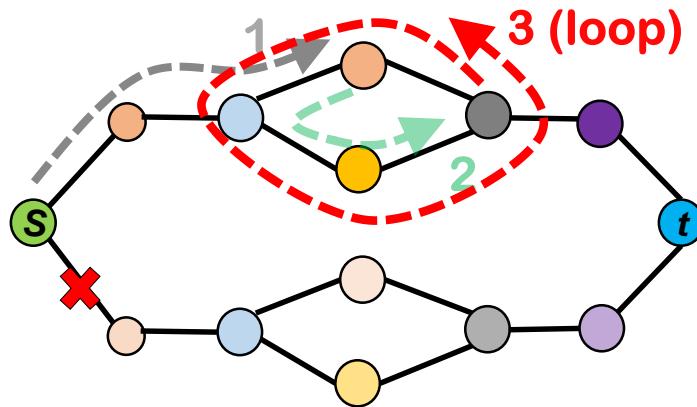
# Impossibility: Via Case Distinction

- Impossible to have 2-resilient *source-matched routing* against dynamic failures *without rewriting bits* even on planar graph
  - In the example, every node must use **link-circular routing** (clockwise or counter-clockwise)
  - Depending on order, fail dynamically: *indistinguishable*



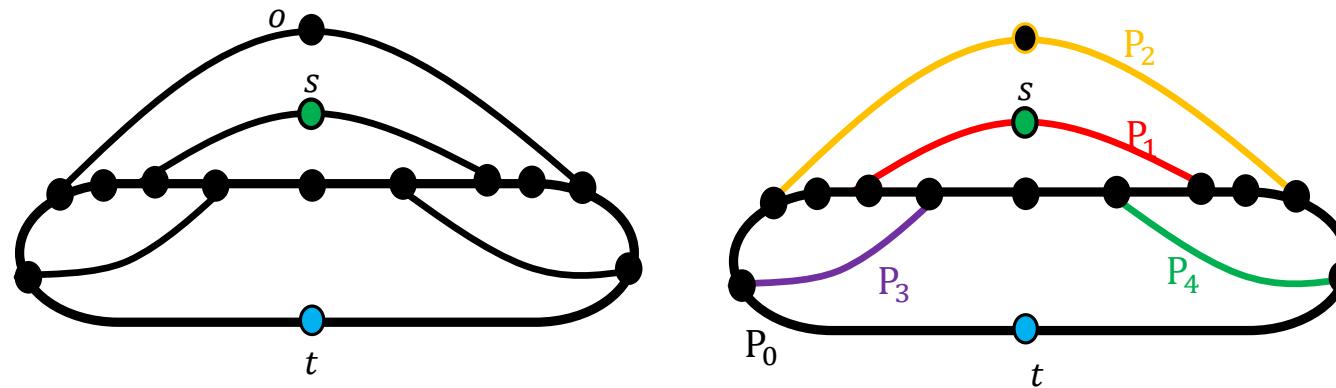
# Impossibility: Via Case Distinction

- Impossible to have 2-resilient *source-matched routing* against dynamic failures *without rewriting bits even on planar graph*
  - In the example, every node must use **link-circular routing** (clockwise or counter-clockwise)
  - Depending on order, fail dynamically: *indistinguishable*



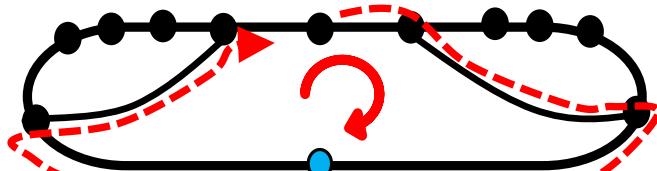
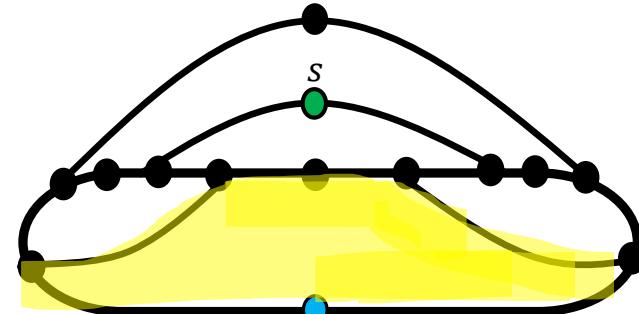
# Dynamic 2-Resilient, 2-Bit Rewriting Algorithm: Via Ear Decomposition

- We can focus on 2-connected graphs wlog
- Ear decomposition of  $G$  is a sequence of subgraphs  $G = P_0 \cup P_1 \dots P_k$ ,
  - $P_0$  is a cycle including a fixed node (destination)  $t$
  - Each  $P_i$  is an ear (path or cycle) with only its endpoints in  $P_0 \cup P_1 \dots P_{i-1}$ : endpoints of each ear belong to earlier ears in the sequence but internal vertices of each ear do not

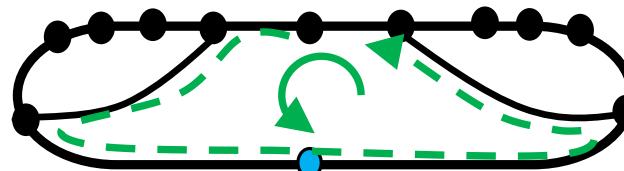


# Directed Routing on Faces

- Idea: Can traverse the boundary in the anti-clockwise and clockwise direction (depending on failures)



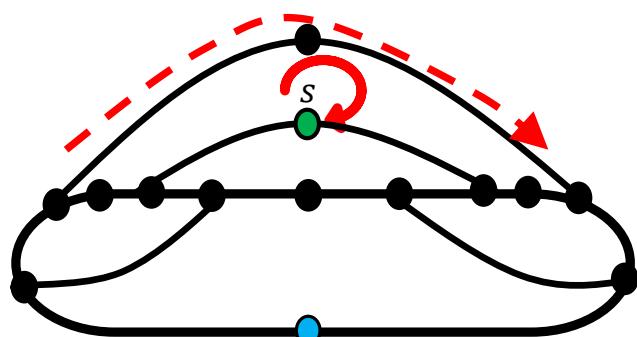
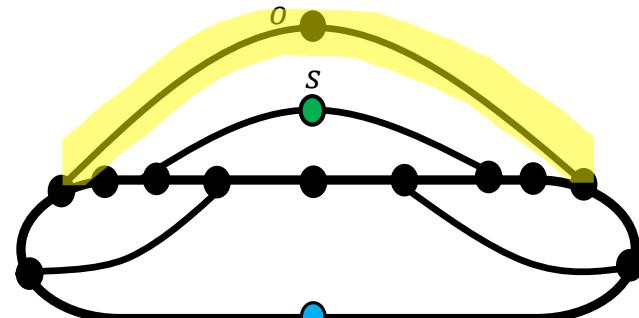
- **Right routing on faces (clockwise)**



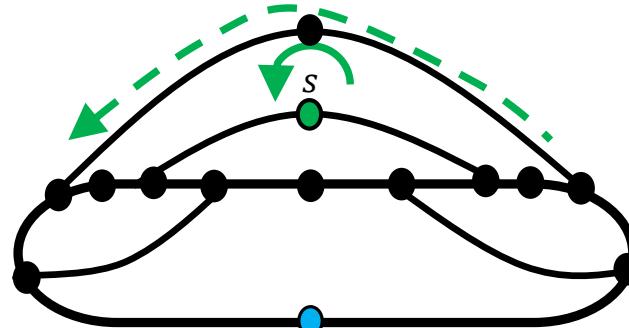
- **left routing on faces (anti-clockwise)**

# Can do this for each ear...

- For each ear, left/right routing traverse it in anti-clockwise/clockwise direction



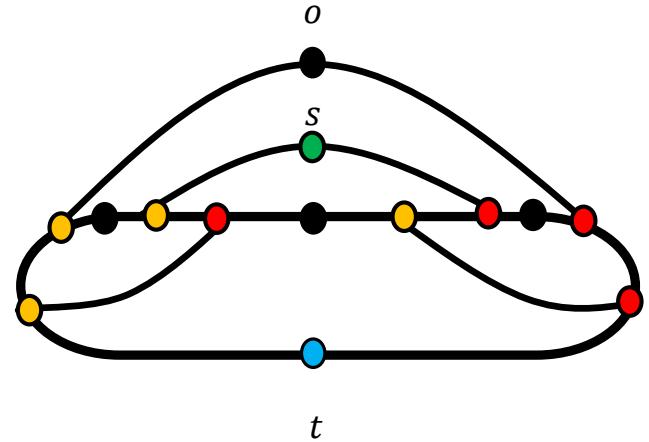
- Right routing on ears (clockwise)



- left routing on ears (anti-clockwise)

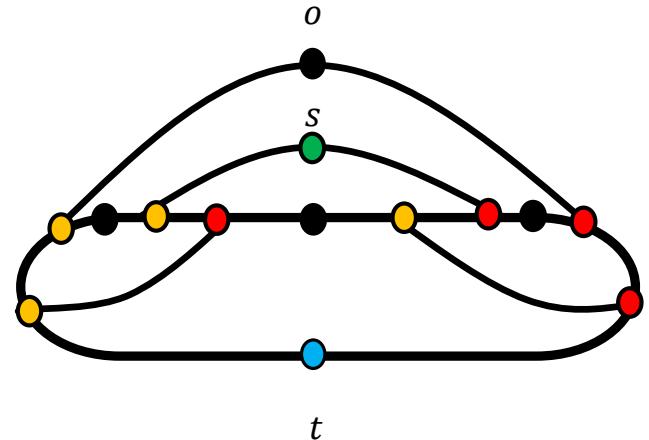
# Idea: Precompute important nodes: left/right-most nodes for each phase

- For each face,  $\exists P_i$  (ear) share common edges
- Precompute: left-most (yellow-colored) and right-most (red-colored)



# Idea: Precompute important nodes: left/right-most nodes for each phase

- For each face,  $\exists P_i$  (ear) share common edges
- Precompute: left-most (yellow-colored) and right-most (red-colored)
- **Complexity:** compute ear-decomposition of 2-edge-connected planar graph and left/right routes on ears and faces, left/right-most nodes: time  $O(n + m)$  per destination

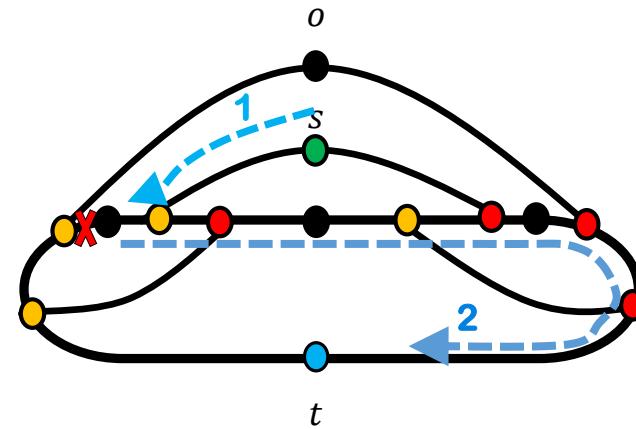


# Idea of the algorithm

- Route packets **through ears** in a "waterfall manner" towards destination. Upon failure, **change direction**.
- If every ear contains **at most one failure**, the packet eventually reaches  $t$
- If a **second failure occurs on the same ear**, routing switches to face-based mode (**1 bit**) on the adjacent faces
- The algorithm first attempts both directions (now using **1 more bit**)
- Will **reach a right/left-most node**

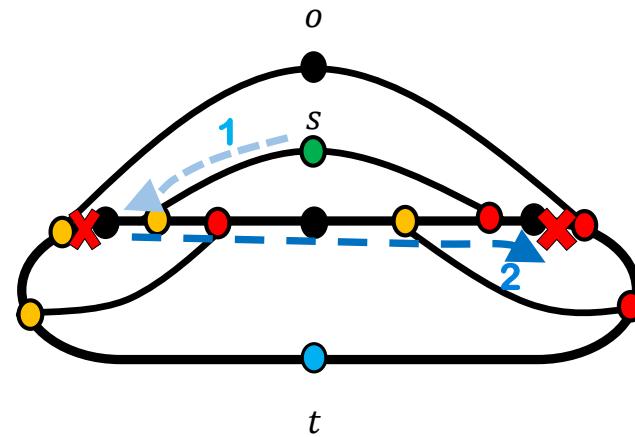
# How to Route by Rewriting One Bit (Ear-Based Routing)

- Starting at the original node, initially  $\beta = 0$
- When bit  $\beta = 0$ , routing on ears,
  - first left and then right direction upon any failure
  - Successful, if seeing  $\leq 1$  failures for ear-based routing



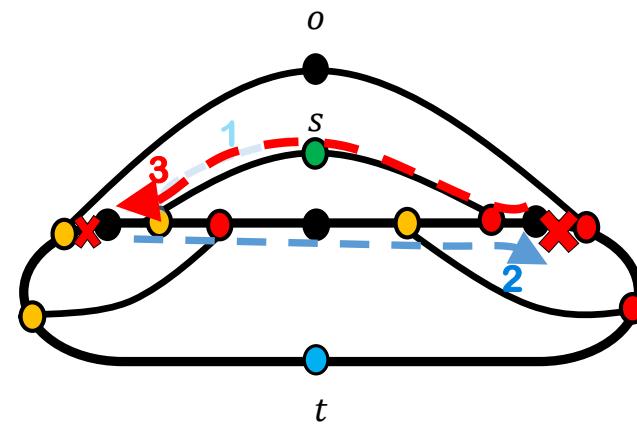
# How to Route by Rewriting One Bit (Ear-Based Routing)

- Starting at the original node, initially  $\beta = 0$
- When bit  $\beta = 0$ , routing on ears,
  - first left and then right direction upon any failure
  - Successful, if seeing  $\leq 1$  failures for ear-based routing
  - Otherwise, when hitting the **second** failure, let  $\beta = 1$



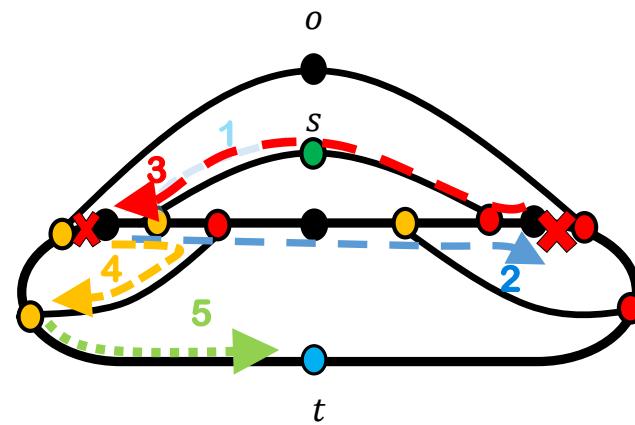
# How to Route by Rewriting One Bit (Face-Based Routing)

- When hitting the **second** failure  $(u, v)$ ,  $\beta = 1$
- Right routing on the face  $f_i$  including  $(v, u)$  until stopping the right-most (red) node ( $\rightarrow 3$ )
- Otherwise, seeing another failure  $(a, b)$  during the right routing on  $f_i$  ( $\rightarrow 3$ )



# How to Route by Rewriting One Bit (Face-Based Routing)

- When hitting the **second** failure  $(u, v)$ ,  $\beta = 1$
- Right routing on the face  $f_i$  including  $(v, u)$  until stopping the right-most (red) node ( $\rightarrow 3$ )
- Otherwise, seeing another failure  $(a, b)$  during the right routing on  $f_i$  ( $\rightarrow 3$ )
- Left routing on the face  $f_j$  including  $(b, a)$  until stopping the left-most (yellow) node ( $\rightarrow 4$ )
- Now, let  $\beta = 0$  to **restart the ear-based routing** ( $\rightarrow 5$ ), always successful (left/right-most node implies left/right routing on the ear)





# Thank you!

Failure type	Per-destination	Per source	Incoming port	Incident links	$k$ -edge-connected graph	Graph Restrictions	Packet rewriting bits	Resiliency	Ref.
static	X		X	X	$k = 2$	planar		No 2-resilient	[Chiesa et al. 2016]
static	X	X	X	X		general		No 3-resilient	[Dai et al. 2023]
static	X	X	X	X		general		$\leq 2$ -resilient	[Dai et al. 2023]
static	X		X	X		general	1 bit	$\geq 2$ -resilient?	Open
static	X	X	X	X		general		$\geq 2$ -resilient?	Open
dynamic	X		X	X	$k = 2$	planar	0	No 2-resilient	[This Talk]
dynamic	X		X	X		general	$\log k$	No $k$ -resilient	[This Talk]
dynamic	X		X	X		planar	1 or 2	$\leq 2$ -resilient	[This Talk]

**Previous results** (highlighted with a red border)

**Our results** (highlighted with a green border)